Method and Apparatus of Arranging Priority Queue and Arbitrating for Memory Access Requests

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

[0001] The present invention generally relates to a method and an apparatus of priority arbitration, and more particularly to a method and an apparatus of arranging priority queue for an arbiter within a memory access interface.

DESCRIPTION OF THE RELATED ART

[0002] As the requirement of consumer media electronic products increasing, in order to enhance the product competition, the concept of SoC (System On Chip, SoC) has become a trend. Due to the device integrated, the chip with SoC concept have less power consumption, greater heat dissipation, and better signal sending quality. Moreover, the concept of SoC is to put more and more devices into one single chip, so each device in this chip have to be integrated or reduced its size in order to meet the concept of SoC. With the integration of devices, the use of devices and substrates can be reduced. Due to the reduction of size of devices, the volume of chip is reduced, and also the package is reduced, so the cost of the chip designed with the concept of SoC can be reduced. One of the most widely used consumer media electronic products is

multi-media player.

[0003] In order to process different video and audio signals, a well-known multi-media player comprises many microprocessors with different functions, for example, to process video signals, a video decoder unit is necessary; to process audio signals, a digital signal processor (DSP) is necessary. Additionally, amounts of memory modules are needed for a well-known multi-media player to handle large data transportation and storage of firmware.

[0004] A memory interface existing between each microprocessor and it's corresponding memory modules is for establishing contact in order to processing the access. To manage requests accessing efficiently, the memory interface comprises an arbiter, arbitrating which one of the access requests asked by the microprocessor has the priority and arbitrating which one of the access requests can use the bus bandwidth. Before arbitrating the priority of bus bandwidth, a standard of bus bandwidth sequence which is called priority queue must be setup. It decides which one of the access requests has the priority to use bus bandwidth based on priority queue when a plurality of access requests are asked at the same time.

[0005] A well-known arrangement of queue is shown in Fig. 1

with a concept of first in first out (FIFO). The first asked access request has the highest priority, which is arranged in forefront (named front in queue) position and must be executed firstly. After finishing executing the forefront access request, the other access requests are pushed forward and a new access request is added into the last position (named rear in queue) of the queue.

[0006] Because of the trend of the concept of SoC (System On Chip) and reduction of memory space in systems, a invention of sharing memory address and data buses is provided earlier, where some parts of system with same functions are integrated or shared. All microprocessors have to access shared memory and data buses via a shard memory interface in the system. Therefore a better method of arranging priority queue and an efficiency arbitrative apparatus is necessary to decide the request priority of a system with amounts of access requests. Furthermore, system performance is increased.

SUMMARY OF THE INVENTION

[0007] The present invention provides a method and an apparatus of priority arbitration deciding which one of the access requests has the priority and which one of the access requests can use bus bandwidth when a large of access requests are asked at the same time without increasing accessing speed

and bus bandwidth.

[0008] The present invention also provides a method of arranging priority queue. When a higher priority access request exists in the highest level priority queue, it prevents system fail if the higher priority access request is not executed at a certain time.

[0009] A preferred embodiment of present invention firstly separates a plurality of access requests into a plurality of priority levels. A plurality of access requests of each priority level are arranged into a priority queue, and the arranging method comprises the following steps: counting cycles and latency of each access request; counting total cycles of all access requests; comparing latency and total cycles of each access request, if latency of an access request is smaller than total cycle, then additionally arranging the same access request into the priority queue once again, else executing the plurality of access requests according to the sequence of priority queue.

[0010] A preferred embodiment of the present invention provides an arbiter, comprising of three arbitrative apparatus which have different priority levels, and each arbitrative apparatus with its own priority level at least comprises of: a plurality of request multiplexers for selecting a plurality of access requests of the priority level; and an ownership selector

for arranging priority queue. In addition, an arbitrative apparatus with its own priority level further comprises of: an OR gate coupled to the arbitrative apparatus of one higher priority level for informing the priority level whether there is any access request is asked, if there is, then arranging the access request into the last position of the priority queue of the higher priority level; and a AND gate coupled to the ownership selector of one lower priority level for asking the lower priority level to point out the next access request when the next access request is at the last position of the priority queue.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] The accompanying drawings are included to provide a further understanding of the invention, and are incorporated in and constitute a part of this specification. The drawings illustrate embodiments of the invention and together with the description, serve to explain the principles of the invention. In the drawings,

[0012] Fig. 1 schematically shows a general purpose way (first in first out, FIFO) of arranging priority queue;

[0013] Fig. 2 schematically shows an executing flow chart of different priority levels in present invention;

[0014] Fig. 3 schematically shows a flow chart of arranging method of priority queue in present invention;

[0015] Fig. 4 schematically shows arbiter architecture of the present invention;

[0016] Fig. 5 schematically shows a high level arbitrative apparatus in present invention;

[0017] Fig. 6 schematically shows a normal level arbitrative apparatus in present invention;

[0018] Fig. 7 schematically shows a low level arbitrative apparatus in present invention;

[0019] Fig. 8 shows a general priority queue arrangement; and

[0020] Fig. 9 shows a priority queue arrangement of the present invention.

DETAILED DESCRIPTION

[0021] The present invention provides a method of arranging priority queue for an arbiter of a memory interface in multi-media player, also provides an apparatus and a method for arbitrating of different priority levels. In this way, when

amounts of access requests are asked at the same time, system finds the one access request which is needed to be executed immediately according to the respected priority of each access request and prevents access request fail or system function fail.

[0022] A preferred embodiment is given as the following to enable one skilled in the art to make and use the present invention as provided within the context of a particular application and its requirements. Various modifications to the preferred embodiment will, however, be apparent to one skilled in the art, and the general principles defined herein may be applied to other embodiments. Therefore the present invention is not intended to be limited to the particular embodiments shown and described herein, but is to be accorded the widest herein disclosed.

[0023] Firstly, separating a plurality of access requests asked from at least one microprocessor into three different priority levels: high priority level, normal priority level and low priority level respectively. A plurality of access requests in each different priority level are arranged as a priority queue and each access request has a corresponding request number. Priority level of an access request can be dynamically adjusted according to the use condition, for example, request access priority of servo controller when playing VCD (Video Compact Disk) is higher than playing DVD (Digital Versatile Disk) due

to the higher frequency of reading disk data; request access priority of MPEG (Moving Picture Experts Group) decoder when playing DVD is higher than playing VCD due to the higher requirements of MPEG decoder.

[0024] Then executing the access requests according to the sequence of high level priority queue, simultaneously the next access request which has the priority is asked and wait. When one access request is in turn to be executed but it is not asked, then executing the next access request. After total access requests of high priority level are once executed, check if there any access request is asked in normal priority level, if there hasn't, then going back to execute high level priority queue; if there has, then executing one access request of normal priority level then going back to high level priority queue. That is to say, when finishing executing a cycle of high priority queue, then executing one access request of normal priority level and going back to execute high level priority queue; after finishing executing a cycle of normal priority queue, then executing one access request of low priority level and going back to execute high level priority queue.

[0025] The flow chart is as the following and shown in Fig. 2:
Assume there has N access requests of high priority level, normal priority level and low priority level respectively,
210 initial value n=0;

220 executing the n-th access request of high level priority queue;

230 estimating if n=N?;

if not, then n=n+1 and go back to step 220;

if yes, then:

240 initial value n'=0;

250 executing the n'-th access request of normal level priority queue;

260 estimating if n'=N?

if not, then n'=n'+1 and go back to step 210;

if yes, then:

270 initial value n"=0; and

280 executing the n"-th access request of low level priority queue, and then back to step 210.

[0026] A well-known method of arranging priority queue is shown in Fig. 8. Take the third access request (slot H2, request No. 10) of high level priority queue as a example, under the worse case in which all of these access requests are asked, the during cycles between executing request No. 10 is at least the total cycles of high level priority queue 26+10+5+13+6+20=80Ts (Ts: tera second, Tera=1e+12). That is to say, request No. 10 must wait for 80Ts then it can be executed again. But it is noticed that the total cycles 80Ts is larger than the latency of request No. 10, 76Ts, therefore, under the worse case, system can't execute request No. 10 within

76Ts, as a results, the request No. 10 will fail. In general to solve this problem, one way is re-arranging one lower priority access request of high level priority queue into normal level priority queue, but it may also cause the access request of normal priority level fail; another way is speeding up accessing speed or increasing bus bandwidth to reduce executing cycles of each access request.

[0027] In the present invention, additional request No. 10 is arranged into the high level priority queue once again (shown in Fig. 9 slot H5). Although the total cycles are increased to 85Ts (26+10+5+13+6+5+20=85), request No. 10 is executed twice in high level priority queue and the during cycles between executing request No. 10 is reduced (shown in Fig. 9, the during cycles are respectively 13+6=19Ts and 20+26+10=56Ts, and both are smaller than new total cycles 85Ts). Therefore, even under the worse case, the access request of request No. 10 is satisfied.

[0028] The method of arranging priority queue in the present invention comprises the following steps: counting latency and cycles of all access requests of each priority queue (cycles means the spending time of executing an access request; latency means the longest waiting time before an access request is accepted, and if the access request is not accepted in its latency, the access request will fail, so that the system function

fail; both units are Ts); counting total cycles of all access requests in priority queue; comparing latency of each access request with total cycles under the worse case respectively, if latency of an access request is smaller than total cycles, then the same access request should be additional arranged into the priority queue once again (the access request is called nonsatisfied access request in which the latency is smaller than total cycles of all access requests in priority queue), else executing access requests according to the sequence of priority queue. However, when an additional access request is arranged in priority queue, then counting the maximum during cycles between the two same access requests; then comparing the latency with the maximum during cycles, if the latency is still smaller than the maximum during cycles, then another the same access request is arranged into the priority again. Repeat this step (recounting maximum during cycles and comparing with the latency between two same access requests), until the nonsatisfied access request can be successfully executed in time. Finally, as some additional access requests are arranged into the priority queue, recounting new total cycle, and comparing with latency again and following the steps above until all access requests in priority queue can be executed successfully during total cycles. Further, the position of the additional access request is the position which has the average during cycles between two same access requests.

[0029] The method of arranging priority queue in present invention is as the following and shown in Fig. 3:

310 counting cycles and latency of each access request;

320 counting total cycles of all access requests in priority queue;

330 comparing that latency of each access request is larger than the total cycles;

If yes, then:

340 executing access requests according to the sequence of priority queue;

If no, then:

350 additionally arranging the nonsatisfied access request into the priority queue once again;

360 counting the maximum during cycles between the two same access requests;

370 comparing whether the latency is larger than the maximum during cycles;

If latency is larger than the maximum during cycles;

If no, then go back to step 350, step 360 and step 370;

If yes, then:

380 recounting new total cycles; and executing the step 330.

[0030] Every time when an additional access request is arranged into the priority queue, the new total cycles are recounted, then comparing with latencies again until all access requests in priority queue can be executing during total cycles.

[0031] As the method of arranging priority queue described above, the method can be used in an arbitrative apparatus. An arbiter of the present invention is shown in Fig. 4, which at least comprises three arbitrative apparatus with different priority levels, a high level arbitrative apparatus 410, a normal level arbitrative apparatus 420, and a low level arbitrative apparatus 430 respectively, wherein each arbitrative apparatus is similar, and a detail description is as the following.

[0032] Refer to Fig. 5, the high level arbitrative apparatus 410 comprises of: a plurality of access request MUXs (multiplexers) 501, each one of the access request MUXs 501 receives all access requests respectively and selects one access request grouped to high priority level from those access requests respectively; a high priority setting register 502 ,setting request number to show which access requests belongs to high priority level; an ownership selector 503, receiving a plurality of access requests which are selected by a plurality of access request MUXs 501, and arranging those selected access requests into a high level priority queue, the ownership selector 503 keeps the last position of high level priority queue for arranging an access request of normal priority level; an ownership MUX 506, finding out the next access request which is waiting for being executed. Furthermore, the ownership selector 503 comprises a next ownership selector unit 504.

When an access request is being executed, an asking point out signal is sent out, then the next ownership selector unit 504 points out the position of the next access request and the ownership MUX 506 finds the corresponding request number from high priority setting register 502 according to the position of high level priority queue.

[0033] Refer to Fig. 6, the normal level arbitrative apparatus 420 comprises of: a plurality of access request MUXs (multiplexers) 601, each one of the access request MUXs 601 receives all access requests respectively and selects one access request grouped to normal priority level from those access requests respectively; a normal priority setting register 602, setting request number to show which access requests belongs to normal priority level; an ownership selector 603, receiving a plurality of access requests which are selected by a plurality of access request MUXs 601, and arranging those selected access requests into a normal level priority queue, the ownership selector 603 keeps the last position of normal level priority queue for arranging an access request of low priority level; an ownership MUX 606, finding out the next access request which is waiting for being executed. Furthermore, the ownership selector 603 comprises a next ownership selector unit 604. When an access request is being executed, an asking point out signal is sent out, then the next ownership selector unit 604 points out the position of the next access request and the

ownership MUX 606 finds the corresponding request number from normal priority setting register 602 according to the position of normal level priority queue.

[0034] Refer to Fig. 7, the low level arbitrative apparatus 430 comprises of : a plurality of access request MUXs (multiplexers) 701, each one of the access request MUXs 701 receives all access requests respectively and selects one access request grouped to low priority level from those access requests respectively; a low priority setting register 702, setting request number to show which access requests belongs to low priority level; an ownership selector 703, receiving a plurality of access requests which are selected by a plurality of access request MUXs 701, and arranging those selected access requests into a low level priority queue; an ownership MUX 706, finding out the next access request which is waiting for being executed. Furthermore, the ownership selector 703 comprises a next ownership selector unit 704. When an access request is being executed, an asking point out signal is sent out, then the next ownership selector unit 704 points out the position of the next access request and the ownership MUX 706 finds the corresponding request number from low priority setting register 702 according to the position of low level priority queue.

[0035] Furthermore, the high level arbitrative apparatus 410 comprises a 2-input AND gate 505 with one input coupled to

the ownership selector 503, estimating if the asking point out signal is sent, and another input coupled to a estimation signal, estimating if the next ownership selector unit 504 points to the last position of the high level priority queue (the position for arranging an access request of normal priority level), and an output coupled to the ownership selector 603 of normal level arbitrative apparatus 420. When an asking point out signal is sent out (true), and the next ownership selector unit 504 points to the last position of high level priority queue (true), then outputting an asking point out signal to normal priority level to ask for pointing out the position of the next access request.

[0036] Furthermore, the normal level arbitrative apparatus 420 comprises an OR gate 607 with multi-inputs coupled to each output of each access request MUXs 601, and an output coupled to the ownership selector 503 of high level arbitrative apparatus 410; when an access request of normal priority level is going to be asked, arranging this access request into the last position of high level priority queue. Additionally, the normal level arbitrative apparatus 420 also comprises a 2-input AND gate 605 with one input coupled to the ownership selector 603, estimating if the asking point out signal is sent, and another input coupled to a estimation signal, estimating if the next ownership selector unit 604 points to the last position of the normal level priority queue (the position for arranging an access request of low priority level), and an output coupled to

the ownership selector 703 of low level arbitrative apparatus 430. When an asking point out signal is sent out (true), and the next ownership selector unit 604 points to the last position of normal level priority queue (true), then outputting an asking point out signal to low priority level to ask for pointing out the position of the next access request.

[0037] Furthermore, the low level arbitrative apparatus 430 comprises an OR gate 707 with multi-inputs coupled to each output of each access request MUXs 701, and an output coupled to the ownership selector 603 of normal level arbitrative apparatus 420; when an access request of low priority level is going to be asked, arranging this access request into the last position of normal level priority queue.

[0038] As the description above, without increasing accessing speed and bus bandwidth, the present invention provides a better method of arranging priority queue and the present invention also provides an apparatus and a method for arbitrating of different priority levels. An arbitrative mechanism with the concept of the present invention can arbitrate large of access requests efficiently, and if there has a higher priority access request in the highest level priority queue, the higher priority access request can be accessed in a short time, thus the access request and system do not fail due to long waiting time.

[0039] Finally, those skilled in the art should appreciate that they can readily use the disclosed conception and specific embodiment as a basis for designing or modifying other structures for carrying put the same purpose of the present invention without departing from the spirit and scope of the invention as defined by the appended claims.